

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

Olivier MIAKINEN

Serial No.: 09/604,919

Filed: June 28, 2000

For: METHOD FOR REMOTE
INTERROGATION OF SNMP AGENTS



Examiner:

Group Art Unit:

Corres. To FR 99/08250

Filed June 28, 1999

McLean, Virginia

**COMPLETION OF
CLAIM FOR BENEFIT OF FILING DATE
OF PRIOR FOREIGN APPLICATION**

Honorable Commissioner of Patents and Trademarks
Washington, DC 20231

Sir:

In the matter of the above-identified application, a claim is hereby made under the provisions of 35 U.S.C. • 119 for the benefit of the filing date of the corresponding French application No. 99 08250 filed June 28, 1999, which is referred to in the Declaration of the present case.

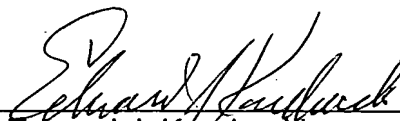
A certified copy of said French application is enclosed herewith.

Respectfully submitted,

Miles & Stockbridge P.C.

Date November 7, 2000

By:


Edward J. Kondracki
Registration No. 20,604

Miles & Stockbridge, P.C.
1751 Pinnacle Drive, Suite 500
McLean, Virginia 22102-3833
Tel.: (703) 903-9000



THIS PAGE BLANK (USPTO)



BULL SA

(2)



BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le **27 JUIN 2000**

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT
NATIONAL DE
LA PROPRIÉTÉ
INDUSTRIELLE

SIEGE
26 bis, rue de Saint Petersburg
75800 PARIS Cédex 08
Téléphone : 01 53 04 53 04
Télécopie : 01 42 93 59 30

THIS PAGE BLANK (USPTO)

REQUÊTE EN DÉLIVRANCE

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 93 59 30

Confirmation d'un dépôt par télécopie ☐

Cet imprimé est à remplir à l'encre noire en lettres capitales

Réservé à l'INPI

DATE DE REMISE DES PIÈCES **28 JUIN 1999**
N° D'ENREGISTREMENT NATIONAL **9908250**
DÉPARTEMENT DE DÉPÔT **75 INPI PARIS B**
DATE DE DÉPÔT **28 JUIN 1999**

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE
À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

BULL S.A.
Patricia BERTRANDIAS / PC 58F35
68, route de Versailles
78434 LOUVECIENNES Cedex
n° du pouvoir permanent - références du correspondant - - - - - téléphone - - - - -
PG 4972 FR3841/PB 01 39.66.66.34

2 DEMANDE Nature du titre de propriété industrielle

☒ brevet d'invention ☐ demande divisionnaire ☐ certificat d'utilité ☐ transformation d'une demande de brevet européen ☐ brevet d'invention ☐ certificat d'utilité n°

demande initiale

Établissement du rapport de recherche

☐ différé ☒ immédiat

Le demandeur, personne physique, requiert le paiement échelonné de la redevance

☐ oui ☒ non

Titre de l'invention (200 caractères maximum)

Procédé d'interrogation à distance d'agents SNMP.

3 DEMANDEUR (S) n° SIREN **6 4 2 0 5 8 7 3 9**

code APE-NAF **3 0 0 C**

Nom et prénoms (souligner le nom patronymique) ou dénomination

BULL S.A.

Forme juridique

S.A.

Nationalité (s) **Française**

Adresse (s) complète (s)

Pays

BULL S.A.
68, route de Versailles
78434 LOUVECIENNES CEDEX

FRANCE

En cas d'insuffisance de place, poursuivre sur papier libre ☐

4 INVENTEUR (S) Les inventeurs sont les demandeurs

☐ oui ☒ non Si la réponse est non, fournir une désignation séparée

5 RÉDUCTION DU TAUX DES REDEVANCES

☐ requise pour la 1ère fois ☐ requise antérieurement au dépôt ; joindre copie de la décision d'admission

6 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE

pays d'origine numéro date de dépôt nature de la demande

7 DIVISIONS

antérieures à la présente demande n°

date

n°

date

8 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE
(nom et qualité du signataire)

Patricia BERTRANDIAS
Salarié BULL S.A.

SIGNATURE DU PRÉPOSÉ À LA RÉCEPTION

SIGNATURE APRÈS ENREGISTREMENT DE LA DEMANDE À L'INPI

BEST AVAILABLE COPY

DÉSIGNATION DE L'INVENTEUR

(si le demandeur n'est pas l'inventeur ou l'unique inventeur)

DEPARTEMENT DES BREVETS

26bis, rue de Saint-Petersbourg

75800 Paris Cédex 08

Tél. : 01 53 04 53 04 - Télécopie : 01 42 93 59 30

FR 3841/PB

N° D'ENREGISTREMENT NATIONAL

99 08 250

TITRE DE L'INVENTION :

Procédé d'interrogation à distance d'agents SNMP.

LE(S) SOUSSIGNÉ(S)

BULL S.A.

DÉSIGNE(NT) EN TANT QU'INVENTEUR(S) (indiquer nom, prénoms, adresse et souligner le nom patronymique) :

Miakinen Olivier
26 rue du Fort
94400 VITRY SUR SEINE
France

NOTA : A titre exceptionnel, le nom de l'inventeur peut être suivi de celui de la société à laquelle il appartient (société d'appartenance) lorsque celle-ci est différente de la société déposante ou titulaire.

Date et signature (s) du (des) demandeur (s) ou du mandataire

Louveciennes, le 28 juin 1999

Bertrandias Patricia

BEST AVAILABLE COPY

DOCUMENT COMPORTANT DES MODIFICATIONS

PAGE(S) DE LA DESCRIPTION OU DES REVENDECATIONS OU PLANCHE(S) DE DESSIN			R.M.*	DATE DE LA CORRESPONDANCE	TAMPON DATEUR DU CORRECTEUR
Modifiée(s)	Supprimée(s)	Ajoutée(s)			
34-38			α	19/1/00	22 MARS 2000 - B B

Un changement apporté à la rédaction des revendications d'origine, sauf si celui-ci découle des dispositions de l'article R.612-36 du code de la Propriété Intellectuelle, est signalé par la mention «R.M.» (revendications modifiées).

BEST AVAILABLE COPY

PROCEDE D'INTERROGATION A DISTANCE D'AGENTS SNMP

La présente invention concerne un procédé d'interrogation à distance d'agents SNMP dans un système informatique.

5

L'art antérieur

Pour administrer une machine sur le réseau Internet à partir d'une autre machine, on utilise le protocole SNMP (Simple Network Management
10 Protocol - Protocole simple de gestion de réseau) de la famille des protocoles TCP/IP. Le protocole SNMP permet à une machine comportant un gestionnaire SNMP de communiquer au travers d'un réseau par un protocole réseau de type SNMP, avec une autre machine comportant un agent SNMP. Pour s'adresser à un agent SNMP à partir d'un gestionnaire
15 SNMP, on utilise un port UDP et une adresse IP. L'agent SNMP gère des objets SNMP caractérisés par des attributs.

Les protocoles d'administration complexes tels que le protocole CMIP
(Common Management Information Protocol - Protocole commun de gestion
20 d'information) permettent de faire des requêtes globales au moyen de filtres complexes. Les filtres complexes se présentent sous la forme d'une expression de recherche booléenne appliquée aux attributs d'objets, à savoir plus précisément un ensemble de conditions sur les attributs (égalité, relations d'ordre, inclusion de chaîne de caractères...) combinées entre elles
25 par tout type d'opérateur (ET, OU, NON, IMPLIQUE...).

Contrairement à de tels protocoles, le protocole SNMP est constitué de trois commandes et d'un message rudimentaires (GET, GETNEXT, SET et TRAP). Il ne connaît que deux types de requêtes permettant la
30 récupération d'informations sur les attributs d'objets : les requêtes GET et GETNEXT. Le protocole SNMP est qualifié de protocole simple par opposition aux protocoles complexes tels que le protocole CMIP. Les

commandes GET et GETNEXT sont des requêtes émises par le gestionnaire SNMP d'une machine sur un objet SNMP d'une autre machine au travers d'un réseau.

5 Un problème se pose lorsqu'un gestionnaire émet une requête complexe du type CMIP en vue de consulter et/ou modifier un objet géré par un agent SNMP. Le protocole SNMP est trop limité pour prendre en charge des opérations complexes de gestion du type CMIP.

10 Actuellement, pour répondre à un tel problème, le système doit parcourir l'ensemble des objets concernés par la requête CMIP et ne conserver que les résultats répondant à ladite requête.

 Or, les ordinateurs étant de plus en plus puissants, le nombre des
15 objets gérés par des agents est de plus en plus grand. Par exemple, de plus en plus d'imprimantes ou de fichiers sont gérés par le même ordinateur. Le nombre de requêtes augmente. Il en résulte une charge importante du réseau, un coût et un temps de réponse accrus.

20 De plus, l'interrogation de nombreuses instances peut entraîner une utilisation intensive du réseau et dégrader les performances de l'ordinateur interrogé.

 Un but de la présente invention consiste à traiter de manière
25 optimisée tout type de requête complexe adressée à un agent SNMP.

Résumé de l'invention

 Dans ce contexte, la présente invention propose un procédé de
30 traitement d'une requête complexe adressée à au moins un agent SNMP d'une machine ressource d'un système informatique à partir d'un

gestionnaire de protocole complexe d'une machine application, chaque agent gérant des tables d'attributs appartenant à la machine ressource, les instances des tables étant référencées par des identificateurs comprenant des index, caractérisé en ce qu'il consiste à :

- 5 • transformer un filtre F1 issu de la requête complexe en un filtre F2 simplifié ne comportant que des conditions sur des index, le filtre F2 respectant les caractéristiques de correspondance suivantes : le filtre F2 laisse passer toutes les requêtes SNMP dont les réponses pourraient vérifier le filtre F1, mais filtre toutes les requêtes SNMP dont les
10 réponses ne peuvent en aucun cas vérifier le filtre F1 ;
- restreindre les requêtes SNMP à celles qui respectent le filtre F2, et appliquer le filtre F1 sur les réponses.

La présente invention concerne également le système de mise en
15 œuvre dudit procédé.

Présentation des figures

D'autres caractéristiques et avantages de l'invention apparaîtront à la
20 lumière de la description qui suit, donnée à titre d'exemple illustratif et non limitatif de la présente invention, en référence aux dessins annexés dans lesquels:

- la figure 1 est une vue schématique d'une forme de réalisation du
25 système selon l'invention ;
- la figure 2 représente partiellement un arbre d'identificateurs d'attributs gérés par le système représenté sur la figure 1 ;
- la figure 3 représente un arbre correspondant à un filtre complexe particulier.

30

Description d'une forme de réalisation de l'invention

Comme le montre la figure 1, le système informatique 1 est distribué et composé de machines 2a, 2b organisées en un ou plusieurs réseaux 3. Une machine 2 est une unité conceptuelle très large, de nature matérielle et
5 logicielle. Les machines peuvent être très diverses, telles que des stations de travail, serveurs, routeurs, machines spécialisées et passerelles entre réseaux. Seuls les composants des machines 2 du système 1 caractéristiques de la présente invention seront décrits, les autres composants étant connus de l'homme du métier.

10

Comme le montre la figure 1, dans la présente invention, le système informatique 1 comprend une machine 2a dite machine application associée à au moins une application et au moins une machine 2b dite machine ressource apte à gérer au moins une ressource. La machine 2a application
15 comporte un gestionnaire 4 de protocole complexe du type CMIP. La machine 2b ressource comporte un agent 5 SNMP. Le gestionnaire 4 dialogue avec un intégrateur d'agent 6 en utilisant le protocole complexe du type CMIP. Le terme 'complexe' sera explicité par la suite. Dans la forme de réalisation illustrée, l'intégrateur d'agent 6 fait partie de la machine 2a
20 application. Toute autre forme de réalisation est susceptible d'être réalisée et par exemple l'intégrateur d'agent 6 pourrait faire partie d'une machine indépendante à la fois de la machine 2a application et de la machine 2b ressource. L'agent 5 SNMP communique avec l'intégrateur d'agent 6 au travers du réseau 3 en utilisant le protocole SNMP. L'intégrateur d'agent 6
25 réalise la traduction du protocole complexe vers le protocole SNMP. Le gestionnaire 4 transmet des requêtes complexes de type CMIP à l'intégrateur d'agent 6 qui les traduit en requêtes simples SNMP envoyées à l'agent 5 SNMP concerné.

30 Il est rappelé qu'un agent SNMP d'une machine 2b ressource gère une MIB, (Management Information Base - Base d'informations

d'administration) organisant sous forme arborescente des attributs SNMP. Un attribut SNMP est ordonné dans un arbre par un identificateur OID (Object Identifier) ; il est d'un type déterminé tel que entier, chaîne de caractères, etc, et présente une valeur à un instant donné. La figure 2
 5 représente un arbre d'identificateurs : l'attribut 'tcpConnState' a pour identificateur 1.3.6.1.2.1.6.13.1.1.

Un agent SNMP peut gérer des objets SNMP globaux et uniques, comme le pourcentage de consommation globale du ou des processeurs de
 10 la machine ressource, et des objets SNMP multiples, comme le pourcentage de consommation de CPU d'une application. Dans ce dernier cas, il y a autant d'instances gérées par l'agent SNMP de ce type d'objet SNMP (pourcentage de consommation de CPU d'une application), qu'il y a d'applications sur la machine. L'objet est dit multi-instancié et est représenté
 15 par une " table SNMP ". Lorsque l'intégrateur d'agent SNMP interroge des tables SNMP comportant de nombreuses instances, il utilise un mécanisme de requêtes et de réponses.

Dans la description qui suit, le terme " attribut " est utilisé pour
 20 désigner un objet SNMP et un attribut CMIS (Common Management Information Service - Service correspondant au protocole CMIP) ; le terme " table " pour désigner une table SNMP et un objet CMIS.

La description qui suit prend pour exemple la table SNMP des
 25 connexions, dénommée tcpConnTable. Comme le montre la figure 2, la table relative aux connexions TCP existantes " tcpConnTable " est désignée par un identificateur, 1.3.6.1.2.1.6.13. 1.3 est fixé par l'ISO. 1.3.6.1 désigne internet. 1.3.6.1.2.1 désigne la MIB standard (mib-II). 1.3.6.1.2.1.6 désigne tcp, 1.3.6.1.2.1.6.13 désigne la table des connexions (la MIB standard mib-II
 30 et la table des connexions TCP sont définies dans la RFC 1213 (RFC -

Request For Comments - Demande de commentaires). La table " tcpConnTable " comprend cinq attributs :

- tcpConnState (OID : 1.3.6.1.2.1.6.13.1.1) ;
- tcpConnLocalAddress (OID : 1.3.6.1.2.1.6.13.1.2) ;
- 5 tcpConnLocalPort (OID : 1.3.6.1.2.1.6.13.1.3) ;
- tcpConnRemAddress (OID : 1.3.6.1.2.1.6.13.1.4) ;
- tcpConnRemPort (OID : 1.3.6.1.2.1.6.13.1.5).

Les attributs tcpConnLocalAddress, tcpConnLocalPort, tcpConnRemAddress, tcpConnRemPort sont des attributs d'index. La concaténation de l'identificateur d'un attribut avec les valeurs, dans l'ordre, de tous les attributs d'index, donne l'identificateur d'une instance particulière de cet attribut. L'annexe 1 donne un exemple de valeurs de ladite table : dans cet exemple, la table comporte 41 instances. Par exemple,

15 l'identificateur de la 37^{ème} instance de l'attribut tcpConnState est :

tcpConnState.219.182.165.53.1021.219.182.165.55.513

soit :

<u>1.3.6.1.2.1.6.13.1.1</u>	<u>. 219.182.165.53</u>	<u>. 1021</u>	<u>. 219.182.165.55</u>	<u>. 513</u>
tcpConnState	1 ^{er} index	2 ^{ème} index	3 ^{ème} index	4 ^{ème}

20 index

identificateur

Par extension, l'identificateur d'une instance désigne également l'ensemble des quatre index, sans le nom de l'attribut. Cet identificateur sert

25 alors à désigner l'instance de n'importe quel attribut de la table.

L'identificateur ci-dessus est celui de la 37^{ème} instance de l'attribut tcpConnState. Le 1^{er} index correspond à la valeur de la 37^{ème} instance de l'attribut d'index tcpConnLocalAddress ; le 2^{ème} index à la valeur de la 37^{ème} instance de l'attribut d'index tcpConnLocalPort ; le 3^{ème} index correspond à

30 la valeur de la 37^{ème} instance de l'attribut d'index tcpConnRemAddress ; le

4^{ème} index à la valeur de la 37^{ème} instance de l'attribut d'index tcpConnRemPort. Ainsi, les quatre derniers composants de l'identificateur de l'attribut tcpConnState correspondent aux valeurs de la même instance des quatre attributs d'index (appelées 1^{er}, 2^{ème}, 3^{ème} et 4^{ème} index).

5

La valeur des attributs d'index est identique au composant de l'identificateur correspondant à l'attribut d'index en question (i^{ème} index). Ainsi, par exemple, la 37^{ème} instance de l'attribut d'index tcpConnLocalAddress dont l'identificateur est :

10 tcpConnLocalAddress.219.182.165.53.1021.219.182.165.55.513 a pour valeur 219.182.165.53 : cette valeur est identique au composant de l'identificateur (1^{er} index) correspondant à l'attribut d'index tcpConnLocalAddress. L'attribut tcpConnState n'est pas un attribut d'index : il est ordonné à l'aide des attributs d'index.

15

Dans cet exemple (annexe 1), l'attribut tcpConnState est de type entier et la première instance présente une valeur égale à 1 à un instant t donné. Dans la forme de réalisation de la figure 1, si le gestionnaire SNMP émet un GET sur la première instance de l'attribut tcpConnState de la machine 2b ressource, l'agent SNMP de la machine 2b ressource répond que la valeur de l'attribut tcpConnState est 1, soit que l'état de la connexion est fermé (closed). Le gestionnaire SNMP peut faire un GETNEXT sur la première instance de l'attribut tcpConnState. L'agent SNMP répond que la valeur recherchée est celle de l'attribut dont l'identificateur

20 (1.3.6.1.2.1.6.13.1.1.0.0.0.0.7.0.0.0.0.0) succède dans un ordre croissant à l'identificateur de la première instance de l'attribut tcpConnState (1.3.6.1.2.1.6.13.1.1.0.0.0.0.0.0.0.0.0.0). Dans le présent exemple, l'attribut, dont l'identificateur succède à l'identificateur de la première instance de l'attribut tcpConnState, est la deuxième instance de l'attribut tcpConnState.

25 L'agent SNMP, installé sur la machine 2b ressource, répond que la valeur de l'instance qui suit, à savoir la deuxième instance de l'attribut tcpConnState

30

est 2 et ainsi de suite sur toutes les instances de la table des connexions jusqu'à ce que l'on passe à un identificateur désignant un autre type d'attribut SNMP tel que, dans cet exemple, tcpConnLocalAdress.

5 La commande GET permet de lire la valeur d'un (ou plusieurs) attribut(s) dont on connaît déjà l'existence (identificateur connu). Seule la commande GETNEXT permet de retrouver d'autres instances en utilisant le fait que toutes les instances sont ordonnées. La commande GETNEXT permet de parcourir toute une table d'instances par des requêtes
10 successives : en partant du début de la table, un premier GETNEXT délivre la première instance existante; ensuite un GETNEXT sur cette première instance donne la deuxième instance, et ainsi de suite jusqu'à la fin de la table.

15 Dans le cas d'une requête complexe relative à une table SNMP, la requête comprend :

- l'identification de la table SNMP considérée ;
- un ensemble de conditions sur des attributs (y compris des index) de la table.

20

Par exemple, une requête complexe sur la table des connexions TCP (tcpConnTable) présente l'ensemble des conditions suivantes :

" le numéro de port distant est 21 et la connexion est active, ou le numéro de port local est supérieur ou égal à 1024 ".

25

L'ensemble des conditions peut être représenté d'une façon un peu plus formelle par un filtre complexe par exemple de type CMIS :

(OU

(ET

30

tcpConnRemPort EGAL_A 21

tcpConnState EGAL_A established(5)

)
 tcpConnLocalPort SUPERIEUR_OU_EGAL_A 1024
).

5 Le filtre complexe peut être représenté sous la forme d'un arbre
 comme le montre la figure 3. Les noeuds de l'arbre qui ne sont pas suivis
 d'autre noeud sont appelés les feuilles. L'arbre de la figure 3 comporte trois
 feuilles : la feuille tcpConnRemPort = 21 ; la feuille tcpConnState =
 established(5) ; la feuille tcpConnLocalPort ≥ 1024. Les feuilles représentent
 10 des conditions sur des attributs tandis que les autres noeuds représentent
 des opérations booléennes sur un ou plusieurs noeuds.

 Une requête complexe dans la présente description est une requête
 transmise par un gestionnaire 4, portant sur des attributs SNMP gérés par
 15 un agent 5 et susceptible d'être représentée par un filtre complexe constitué
 d'un nombre quelconque de conditions sur un nombre quelconque
 d'attributs, reliées entre elles par un nombre quelconque d'opérateurs tels
 que ET, OU, NON, OU-EXCLUSIF, etc.

20 Le protocole SNMP ne permet pas à l'agent 5 SNMP de répondre à
 de telles requêtes complexes, compte tenu des commandes rudimentaires
 qui constituent le protocole SNMP, comme vu plus haut.

 Le procédé selon la présente invention consiste à traiter ladite
 25 requête complexe au moyen de l'intégrateur d'agent 6 qui traduit ladite
 requête complexe (dans l'exemple illustré de type CMIP) en requêtes SNMP,
 et à optimiser le nombre de requêtes SNMP transmises sur le réseau 3, en
 particulier le nombre des requêtes GETNEXT.

30 Selon la présente invention, le procédé de traitement d'une requête
 complexe adressée à au moins un agent 5 SNMP d'une machine 2b

ressource du système informatique 1 à partir d'un gestionnaire de protocole complexe d'une machine 2a application, chaque agent 5 gérant des tables d'attributs appartenant à la machine 2b ressource, les instances des tables étant référencées par des identificateurs comprenant des index, consiste à :

- 5 • transformer un filtre F1 issu de la requête complexe en un filtre F2 simplifié ne comportant que des conditions sur des index, le filtre F2 respectant les caractéristiques de correspondance suivantes : le filtre F2 laisse passer toutes les requêtes SNMP dont les réponses pourraient vérifier le filtre F1, mais filtre toutes les requêtes SNMP dont les
10 réponses ne peuvent en aucun cas vérifier le filtre F1 ;
- restreindre les requêtes SNMP à celles qui respectent le filtre F2, et appliquer le filtre F1 sur les réponses.

Le procédé selon la présente invention comprend les étapes
15 suivantes :

- 1- transformer un filtre F1 complexe issu de la requête complexe en un filtre F2 simplifié ne comportant que des conditions sur des index et respectant les caractéristiques de correspondance ;
- 2- déterminer la première instance potentielle vérifiant les conditions du filtre
20 F2 simplifié ; l'identificateur qui est juste inférieur à l'identificateur de l'instance potentielle déterminée est appelé identificateur de test ;
- 3- retrouver à partir d'une requête SNMP l'instance de la table ayant pour identificateur celui qui suit l'identificateur de test. Si aucune instance n'est retrouvée, le procédé de traitement est terminé. Si une instance est
25 retrouvée, l'instance retrouvée est appelée instance solution ;
- 4- appliquer le filtre F1 complexe à l'instance solution ; si l'instance vérifie le filtre F1, elle fait partie de la réponse à la requête complexe traitée ;
- 5- déterminer la première instance potentielle dont l'identificateur est supérieur à l'identificateur de l'instance solution et vérifiant les conditions du
30 filtre F2 simplifié. Si aucune instance n'est retrouvée, le procédé de traitement est terminé. Si une instance est retrouvée, l'identificateur qui est

juste inférieur à l'identificateur de l'instance potentielle est appelé identificateur de test et le procédé reprend à partir de la troisième étape.

La première étape consiste à construire à partir d'un filtre F1 issu
5 d'une requête complexe donnée, dans l'exemple illustré de type CMIP, un filtre F2 simplifié qui ne comprend que des conditions portant sur des index.

Le filtre simplifié F2 respectent les caractéristiques, appelées caractéristiques de correspondance, suivantes:

- 10 - s'il peut exister des valeurs d'attributs telles qu'une instance donnée de la table vérifie le filtre F1, alors cette instance vérifie le filtre F2;
- si une instance de la table ne peut pas vérifier le filtre F1 quelles que soient les valeurs d'attributs qui ne sont pas des index, alors cette instance ne vérifie pas le filtre F2.

15

Si aucune instance ne vérifie de prime abord le filtre complexe F2, l'intégrateur d'agent 6 transmet une réponse vide à la requête complexe transmise par le gestionnaire 4. Selon une forme de réalisation, la dite vérification est réalisée de la manière suivante : l'intégrateur d'agent 6
20 dispose d'une liste de règles définissant les filtres qui ne sont vérifiés de prime abord par aucune instance, telles que par exemple :

" Si un filtre porte sur une valeur d'attribut V du type " $V=a$ ET $V>b$ " et si " $a\leq b$ " alors le filtre n'a pas de réponse". C'est le cas par exemple lorsque V représente un numéro de port distant, $a=22$ et $b=41$.

25

Une autre forme de réalisation sera décrite ultérieurement.

L'intégrateur d'agent 6 peut également disposer d'une liste de règles de transformation telles que par exemple :

- ($V\geq a$) ET ($V\leq a$) correspond à $V=a$;
- 30 ($V\geq a$) ET (V différent de a) correspond à $V>a$;
- ($V\geq a$) OU ($V<a$) correspond à VRAI (condition toujours vérifiée).

En partant d'un filtre complexe quelconque F1, le procédé consiste, au moyen de l'intégrateur d'agent 6, à obtenir un filtre simplifié F2 de la forme suivante :

5

(OU

(ET

condition sur le 1^{er} index

condition sur le 2^{ème} index

10

...

condition sur le n^{ième} index

)

(ET

condition sur le 1^{er} index

15

condition sur le 2^{ème} index

...

condition sur le n^{ième} index

)

...

20

)

Si le filtre complexe contient d'autres opérateurs que des ET, des OU et des NON, le procédé selon l'invention transforme lesdits opérateurs en combinaisons de ET, OU et NON à l'aide de règles connues sur les opérateurs logiques.

25

Par exemple pour les opérateurs binaires OU-EXCLUSIF et IMPLIQUE:

(OU-EXCLUSIF A B) est équivalent à (OU (ET A (NON B)) (ET B (NON A))) ;

30 (IMPLIQUE A B) est équivalent à (OU (NON A) B).

Le procédé consiste ensuite à repousser tous les opérateurs NON vers les feuilles de l'arbre représentant le filtre en appliquant les règles suivantes autant de fois qu'il est possible de le faire :

- remplacer (NON (OU A B C ...)) par (ET (NON A) (NON B) (NON C) ...)
- 5 - remplacer (NON (ET A B C ...)) par (OU (NON A) (NON B) (NON C) ...)
- remplacer (NON (NON A)) par A.

L'étape suivante consiste à supprimer du filtre toutes les conditions portant sur des attributs qui ne sont pas des index. Il est à noter que, si X est
 10 une condition portant sur un attribut qui n'est pas un index, alors il peut exister des valeurs de l'attribut pour lesquelles X est vrai, et d'autres pour lesquelles (NON X) est vrai : tous les (NON X), repoussés vers les feuilles dans l'étape précédente, sont remplacés par la constante VRAI, puis tous les X restants sont remplacés par la constante VRAI.

15

Cette étape permet d'assurer les caractéristiques de correspondance.

Le filtre est à nouveau simplifié en appliquant les règles suivantes autant de fois qu'il est possible de le faire :

- 20 • Remplacer toutes les opérations ET ne contenant que des opérandes VRAI par la constante VRAI.

Par exemple, remplacer (ET VRAI VRAI VRAI) par VRAI.

- Retirer tous les opérandes VRAI des autres opérations ET.

Par exemple, remplacer (ET A B VRAI C VRAI) par (ET A B C).

- 25 • Remplacer toutes les opérations OU contenant au moins un opérande VRAI par la constante VRAI.

Par exemple, remplacer (OU A B VRAI C VRAI) par VRAI.

- Remplacer les opérations ET et OU à un seul opérande par cet opérande.

Par exemple, remplacer (ET A) par A.

- 30 • Factoriser les ET et OU imbriqués.

Par exemple, remplacer (OU A (OU B C) D) par (OU A B C D).

- Regrouper les conditions concernant le même index. Dans la description qui suit, des conditions, simples ou complexes, portant respectivement sur l'index numéro 1, 2, 3, ... n seront appelées C1, C2, C3, ... Cn. De la même manière, une condition quelconque portant sur un index quelconque k, compris entre 1 et n sera appelée Ck.
- 5 Une condition du type $(OU (ET C_k C_k) C_k (NON C_k))$ est remplacée par une seule condition de type Ck un peu plus complexe. Ce type de transformation aura notamment pour effet de supprimer tous les opérateurs NON.
- 10 • Lorsqu'une condition de type Ck est toujours vraie ou toujours fausse, remplacer cette condition par la valeur VRAI ou FAUX, respectivement.
- Selon une forme de réalisation du système 1, l'intégrateur d'agent 6 dispose d'une liste de règles telles que par exemple :
- 15 (index 3 < 22) ET (index 3 > 23) correspond à une condition FAUX ;
- (index 3 < 22) OU (index 3 > 21) correspond à une condition VRAI.
- Remplacer toutes les opérations OU ne contenant que des conditions FAUX par la constante FAUX.
- Par exemple, remplacer $(OU FAUX FAUX FAUX)$ par FAUX.
- Retirer toutes les conditions FAUX des autres opérations OU.
- 20 Par exemple, remplacer $(OU A B FAUX C FAUX)$ par $(OU A B C)$.
- Remplacer toutes les opérations ET contenant au moins une condition FAUX par la constante FAUX.
- Par exemple, remplacer $(ET A B FAUX C FAUX)$ par FAUX.
- 25 Les règles sont appliquées dans cet ordre ou dans un ordre différent jusqu'à ce que plus aucune desdites règles ne soit applicable.

Les règles ci-dessus permettent d'obtenir un filtre simplifié appartenant à l'un des trois cas suivants:

- 30 1) Le filtre se résume à la seule condition VRAI ;
- 2) Le filtre se résume à la seule condition FAUX ;

3) Le filtre ne contient que des opérateurs ET et OU, chacun comprenant au maximum une condition sur chacun des index, et aucun ne comprenant la valeur VRAI ou FAUX.

5 Dans le premier cas, aucune connaissance n'est susceptible d'être obtenue a priori sur les instances qui conviendront : la table doit être parcourue dans son intégralité.

Dans le second cas, aucune instance ne peut convenir : l'agent
10 intégrateur 6 répond au filtre complexe par une réponse vide.

Dans le troisième cas, le procédé consiste à regrouper tous les OU à la racine du filtre : tout opérateur OU contenu dans un opérateur ET est développé.

15 Par exemple,
(ET C3 (OU C1 C2) C6) devient (OU (ET C3 C1 C6) (ET C3 C2 C6))

Certaines des simplifications précédentes doivent à nouveau être appliquées, à savoir :

- Remplacer les opérations ET et OU à un seul opérande par cet opérande.
- 20 • Factoriser les ET et OU imbriqués.
- Regrouper les conditions concernant le même index.
- Lorsqu'une condition de type Ck est toujours vraie ou toujours fausse, remplacer cette condition par la valeur VRAI ou FAUX, respectivement.
- Remplacer toutes les opérations ET contenant au moins une condition
25 FAUX par la constante FAUX.
- Remplacer toutes les opérations ET ne contenant que des conditions VRAI par la constante VRAI.
- Retirer toutes les conditions VRAI des autres opérations ET.
- Remplacer toutes les opérations OU contenant au moins une condition
30 VRAI par la constante VRAI.

- Remplacer toutes les opérations OU ne contenant que des conditions FAUX par la constante FAUX.
- Retirer toutes les conditions FAUX des autres opérations OU.

5 Les règles sont appliquées dans cet ordre ou dans un ordre différent jusqu'à ce que plus aucune desdites règles ne soit applicable.

Le filtre simplifié obtenu appartient à l'un des cas suivants:

- 1) Le filtre se résume à la seule condition VRAI ;
- 10 2) Le filtre se résume à la seule condition FAUX ;
- 3) Le filtre se résume à une seule condition C_k ;
- 4) Le filtre se présente sous la forme d'un ET entre deux ou plusieurs conditions C_k ;
- 5) Le filtre se présente sous la forme d'un OU entre deux ou
- 15 plusieurs filtres du type 3) ou 4).

En dehors des cas 1) et 2) déjà traités plus haut, le filtre simplifié F2 a donc la forme suivante (ou bien une forme plus simple que la forme suivante, qui peut facilement s'y ramener) :

$$20 \quad F2 = (OU (ET C1_{(1)} C2_{(1)} \dots Cn_{(1)}) \dots (ET C1_{(n)} C2_{(n)} \dots Cn_{(n)}) \dots).$$

Le filtre simplifié F2 peut s'écrire sous la forme suivante:

$$F2 = (OU F2_{(1)} F2_{(2)} \dots F2_{(n)} \dots F2_{(m)})$$

où chaque $F2_{(n)}$ a la forme suivante:

$$25 \quad F2_{(n)} = (ET C1_{(n)} C2_{(n)} \dots Cn_{(n)})$$

La deuxième étape consiste, au moyen de l'intégrateur d'agent 6, à déterminer la première instance, appelée instance potentielle, qui vérifie les conditions du filtre simplifié F2.

La première instance potentielle vérifiant le filtre F2 est la plus petite des premières instances potentielles dites "locales" vérifiant chacun des $F2_0$: le procédé consiste donc à rechercher la première instance qui vérifie un filtre du type $(ET\ C1_0\ C2_0\ \dots\ Cn_0)$. Pour tout i , l'identificateur de la

5 première instance potentielle "locale" vérifiant $F2_0$ est obtenu en concaténant la première valeur $I1_0$ vérifiant $C1_0$ à la première valeur $I2_0$ vérifiant $C2_0$, etc. jusqu'à In_0 vérifiant Cn_0 . L'identificateur de l'instance potentielle locale, dite instance potentielle locale "zéro" parce que instance potentielle locale obtenue la première en début de procédé, est

10 $I1_0.I2_0.\dots.In_0$. S'il n'y a pas de condition sur un index k ($k^{ième}$ index) donné, la première valeur possible est prise pour cet index : 0 pour un entier, 0.0.0.0 pour une adresse IP, etc. L'identificateur de l'instance potentielle est le plus petit des identificateurs des instances potentielles locales zéro obtenues.

15

Il est à noter que l'intégrateur d'agent 6 connaît les valeurs des index (comme vu plus haut) contrairement aux valeurs des attributs pour lesquels il doit interroger l'agent 5 qui gère l'attribut concerné.

20 Pour un indice i donné et pour chaque indice k , il existe au moins une valeur Ik_0 qui vérifie la condition Ck_0 , sinon la condition Ck_0 aurait été supprimée vers la fin de la première étape. L'autre forme de réalisation du système mentionnée plus haut et permettant à l'intégrateur d'agent de détecter les filtres pour lesquels aucune réponse n'est possible est la

25 suivante : attendre la $2^{ème}$ étape pour les supprimer et dans la $2^{ème}$ étape, s'il existe une condition Ck pour laquelle aucune valeur n'est retrouvée, la condition Ck fait partie des conditions pour lesquelles aucune réponse n'est possible.

30 L'identificateur qui est juste inférieur à celui de l'instance potentielle est appelé identificateur de test. Il est obtenu de la manière suivante :

- Si le dernier chiffre de l'identificateur de l'instance potentielle est différent de 0, l'identificateur de test est identique à l'identificateur de l'instance potentielle sauf au niveau du dernier chiffre qui est juste inférieur au dernier chiffre de l'identificateur de l'instance potentielle.

5 Par exemple:

195.3.27.2 (identificateur de l'instance potentielle) -> 195.3.27.1
(identificateur de test)

- Si le dernier chiffre de l'identificateur de l'instance potentielle est égal à 0, l'identificateur de test correspond à l'identificateur de l'instance

10 potentielle sans son dernier chiffre.

Par exemple:

195.3.27.0 (identificateur de l'instance potentielle) -> 195.3.27
(identificateur de test).

15 L'intégrateur d'agent 6 applique, dans une troisième étape, la requête GETNEXT sur l'identificateur de test.

Si la réponse au GETNEXT indique que la fin de la table est atteinte, l'intégrateur d'agent 6 répond à la requête complexe qu'il n'y a plus d'autre

20 réponse. Le traitement est terminé : ceci constitue le premier cas de terminaison.

Si une instance est obtenue, elle est appelée instance solution. Si l_k désigne la valeur de chacun des index k de l'instance solution,

25 l'identificateur de l'instance solution, appelé identificateur solution, est : $l_1.l_2...l_n$.

Dans une quatrième étape, l'intégrateur d'agent 6 applique le filtre F1 complexe à la dite instance solution. Si la réponse convient, elle est

30 renvoyée en réponse à la requête complexe. Que la réponse convienne ou non, le procédé poursuit le traitement.

Le procédé, dans la cinquième étape, au moyen de l'intégrateur d'agent 6, détermine la première instance potentielle dont l'identificateur est strictement supérieur à l'identificateur solution et qui vérifie le filtre simplifié

5 F2. Cette première instance potentielle est la plus petite des premières instances potentielles locales pour chaque filtre $F2_{(i)}$, dont l'identificateur est strictement supérieur à l'identificateur solution et qui vérifie le filtre $F2_{(i)}$.

11.12. ... I_n est l'identificateur solution, comme vu plus haut.

10

Les opérations suivantes sont effectuées pour chaque $F2_{(i)}$.

Soit p le premier indice tel que I_p ne vérifie pas la condition $Cp_{(i)}$.

S'il n'existe pas un tel indice p (c'est-à-dire si l'instance vérifie complètement le filtre), alors on prend $p = n$.

15

Le procédé effectue la boucle suivante tant que l'indice $p > 0$ et qu'aucune instance n'a été trouvée :

{ — S'il existe un $Jp_{(i)} > I_p$ qui vérifie la condition $Cp_{(i)}$, alors l'identificateur de l'instance potentielle locale est constitué de la manière

20 suivante:

- pour tout index $k < p$, on prend la valeur I_k ;

- pour l'index p , on prend la valeur $Jp_{(i)}$;

- pour tout index $k > p$, on prend la valeur $I_{k-0_{(i)}}$;

et le procédé quitte la boucle ;

25

Sinon $p := p-1$ et le procédé continue à boucler avec la nouvelle valeur de p ;

}

Il est rappelé que la valeur $I_{k-0_{(i)}}$ correspond à l'identificateur de

30 l'instance potentielle locale zéro en début de procédé ($I1_{-0_{(i)}}, I2_{-0_{(i)}}, \dots, I_n_{-0_{(i)}}$) (voir plus haut).

Si $p = 0$, c'est qu'il n'y a plus d'autre instance qui vérifie cet ensemble de conditions pour $F2_0$.

5 Si aucune instance potentielle locale n'est obtenue (c'est-à-dire si le procédé quitte la boucle avec $p=0$ pour chaque $F2_0$), c'est qu'il n'y a plus d'autre instance qui vérifie le filtre simplifié $F2$; l'intégrateur d'agent 6 indique en réponse à la requête complexe du gestionnaire 4 qu'il n'y a plus d'autre réponse. Le traitement est terminé : ceci constitue le deuxième cas
10 de terminaison.

Si au moins une instance potentielle locale est obtenue, l'identificateur de l'instance potentielle est le plus petit des identificateurs des instances potentielles locales obtenues. L'identificateur qui est juste
15 inférieur à l'identificateur de l'instance potentielle est appelé identificateur de test. Le procédé reprend à partir de la troisième étape. L'intégrateur 6 d'agent applique un GETNEXT sur l'identificateur de test et ainsi de suite.
- Les troisième, quatrième et cinquième étapes sont appliquées jusqu'à ce que le traitement se termine dans l'un des deux cas de terminaison
20 présentés précédemment.

Le procédé selon l'invention est décrit au travers de l'exemple détaillé qui suit. Les valeurs de la table de connexion sont données par l'annexe 1.

25 Le filtre complexe $F1$ est le suivant :

(ET

tcpConnState EGAL_A 5

(OU

tcpConnLocalPort EGAL_A 21

30 tcpConnLocalPort EGAL_A 23

)

```

(NON
    tcpConnRemAddress EGAL_A 127.0.0.1
)
tcpConnRemPort SUPERIEUR_OU_EGAL_A 1024
5      )

```

La première étape est automatique dans cet exemple. La condition sur l'attribut tcpConnState est mis à la valeur VRAI et de ce fait, le filtre simplifié F2 est de la forme :

10 $F2 = (OU\ F2_{(1)}) = F2_{(1)} = (ET\ C1_{(1)}\ C2_{(1)}\ C3_{(1)}\ C4_{(1)})$

avec :

$C1_{(1)}$: aucune contrainte sur le premier index

$C2_{(1)}$: deuxième index EGAL_A 21 ou 23

$C3_{(1)}$: troisième index DIFFERENT_DE 127.0.0.1

15 $C4_{(1)}$: quatrième index SUPERIEUR_OU_EGAL_A 1024

La deuxième étape consiste à rechercher l'instance potentielle locale zéro, à savoir les premières valeurs possibles pour chacune des conditions.

Les premières valeurs possibles sont :

20 $I1_0_{(1)} = 0.0.0.0$

$I2_0_{(1)} = 21$

$I3_0_{(1)} = 0.0.0.0$

$I4_0_{(1)} = 1024.$

25 L'identificateur de l'instance potentielle locale zéro est obtenu en concaténant la première valeur vérifiant $C1_{(1)}$ à la première valeur vérifiant $C2_{(1)}$ à la première valeur vérifiant $C3_{(1)}$ et à la première valeur vérifiant $C4_{(1)}$: $I1_0_{(1)}.I2_0_{(1)}.I3_0_{(1)}.I4_0_{(1)}$, soit 0.0.0.0.21.0.0.0.0.1024. Comme il n'existe qu'un unique $F2_0$, à savoir $F2_{(1)}$, l'identificateur obtenu est le plus

30 petit et correspond à l'identificateur de l'instance potentielle. L'identificateur de test est donc 0.0.0.0.21.0.0.0.0.1023.

La troisième étape consiste à appliquer la première requête GETNEXT sur cet identificateur de test. Le résultat obtenu (identificateur solution I1.I2.I3.I4) est 0.0.0.0.23.0.0.0.0.

5

Dans une quatrième étape, le filtre F1 est appliqué audit résultat, à savoir à l'identificateur solution. Le filtre F1 échoue sur cette instance ($I4 < 1024$ et tcpConnState égal à 2 différent de 5).

10 Dans une cinquième étape, la première instance potentielle locale dont l'identificateur est strictement supérieur à l'identificateur solution (identificateur de l'instance solution) est calculé. Soit p le premier indice tel que I_p ne vérifie pas la condition $C_{p(1)}$.

Ce n'est pas $p=1$ car il n'y a pas de contrainte sur $I1$;

15 Ce n'est pas $p=2$ car $I2=23$ vérifie $C_{2(1)}$;

Ce n'est pas $p=3$ car $I3=0.0.0.0$ vérifie $C_{3(1)}$;

C'est $p=4$ car $I4=0$ ne vérifie pas $C_{4(1)}$.

20 Le procédé rentre dans la boucle de calcul (le procédé reste dans la boucle tant que $p > 0$ et qu'aucune instance n'a été trouvée) :

$p=4$: existe-t-il un $J_{4(1)}$ supérieur à $I4=0$ qui vérifie $C_{4(1)}$? Oui :

$J_{4(1)}=1024$

Le nouvel identificateur de l'instance potentielle est donc I1.I2.I3.J4(1), soit 0.0.0.0.23.0.0.0.0.1024. Le nouvel identificateur de test est alors

25 0.0.0.0.23.0.0.0.0.1023. Le procédé reprend à la troisième étape : le résultat de la deuxième requête GETNEXT sur cet identificateur de test donne 0.0.0.0.25.0.0.0.0.0 (identificateur solution I1.I2.I3.I4). Le filtre F1 échoue sur cette instance ($I4 < 1024$, I2 différent de 21 ou 23 et tcpConnState égal à 2 différent de 5).

30

Le procédé continue en calculant la première instance potentielle locale dont l'identificateur est strictement supérieur à l'identificateur solution (identificateur de l'instance solution). Soit p le premier indice tel que I_p ne vérifie pas la condition $C_{p(1)}$.

- 5 Ce n'est pas $p=1$ car il n'y a pas de contrainte sur I_1 ;
C'est $p=2$ car $I_2=25$ ne vérifie pas $C_{2(1)}$.

Le procédé rentre dans la boucle de calcul (le procédé reste dans la boucle tant que $p>0$ et qu'aucune instance n'a été trouvée) :

- 10 $p=2$: existe-t-il un $J_{2(1)}$ supérieur à $I_2=25$ qui vérifie $C_{2(1)}$? Non.
 $p=1$: existe-t-il un $J_{1(1)}$ supérieur à $I_1=0.0.0.0$? Oui : $J_{1(1)}=0.0.0.1$

Le nouvel identificateur de l'instance potentielle est donc $J_{1(1)}.I_{2_0(1)}.I_{3_0(1)}.I_{4_0(1)}$, soit $0.0.0.1.21.0.0.0.0.1024$ (il est rappelé que l'identificateur de l'instance potentielle locale zéro en début de procédé
15 $I_{1_0(1)}.I_{2_0(1)}.I_{3_0(1)}.I_{4_0(1)}$ est égal à $0.0.0.0.21.0.0.0.0.1024$). Le nouvel identificateur de test est alors $0.0.0.1.21.0.0.0.0.1023$. Le résultat du troisième GETNEXT sur cet identificateur de test donne $127.0.0.1.1026.127.0.0.1.2600$ (identificateur solution $I_1.I_2.I_3.I_4$). Le filtre F_1 échoue sur cet identificateur solution (I_2 différent de 21 ou 23).

20

Le procédé continue en calculant la première instance potentielle locale dont l'identificateur est strictement supérieur à l'identificateur solution. Soit p le premier indice tel que I_p ne vérifie pas la condition $C_{p(1)}$.

- Ce n'est pas $p=1$ car il n'y a pas de contrainte sur I_1 ;
25 C'est $p=2$ car $I_2=1026$ ne vérifie pas $C_{2(1)}$.

Le procédé rentre dans la boucle de calcul (le procédé reste dans la boucle tant que $p>0$ et qu'aucune instance n'a été trouvée) :

- $p=2$: existe-t-il un $J_{2(1)}$ supérieur à $I_2=1026$ qui vérifie $C_{2(1)}$? Non.
30 $p=1$: existe-t-il un $J_{1(1)}$ supérieur à $I_1=127.0.0.1$? Oui :
 $J_{1(1)}=127.0.0.2$

Le nouvel identificateur de l'instance potentielle est donc $J1_{(1)}.I2_0_{(1)}.I3_0_{(1)}.I4_0_{(1)}$, soit 127.0.0.2.21.0.0.0.0.1024. Le nouvel identificateur de test est alors 127.0.0.2.21.0.0.0.0.1023. Le résultat du quatrième GETNEXT sur cet identificateur de test donne

5 219.182.165.53.23.219.182.165.55.4109 (identificateur solution I1.I2.I3.I4).
Le filtre F1 réussit sur cet identificateur solution ; une réponse est donc envoyée à la requête complexe.

Le procédé continue en calculant la première instance potentielle

10 locale dont l'identificateur est strictement supérieur à l'identificateur solution.
Soit p le premier indice tel que I_p ne vérifie pas la condition $Cp_{(1)}$.
Ce n'est pas $p=1$ car il n'y a pas de contrainte sur I1 ;
Ce n'est pas $p=2$ car $I2=23$ vérifie $C2_{(1)}$;
Ce n'est pas $p=3$ car $I3=219.182.165.55$ vérifie $C3_{(1)}$;
15 Ce n'est pas $p=4$ car $I4=4109$ vérifie $C4_{(1)}$.
Puisqu'il n'existe pas de tel p , on prend $p=n=4$.

Le procédé rentre dans la boucle de calcul (le procédé reste dans la boucle tant que $p>0$ et qu'aucune instance n'a été trouvée) :

20 $p=4$: existe-t-il un $J4_{(1)}$ supérieur à $I4=4109$ qui vérifie $C4_{(1)}$? Oui :
 $J4_{(1)}=4110$
Le nouvel identificateur de l'instance potentielle est donc I1.I2.I3. $J4_{(1)}$, soit 219.182.165.53.23.219.182.165.55.4110. Le nouvel identificateur de test est alors 219.182.165.53.23.219.182.165.55.4109. Le résultat du cinquième

25 GETNEXT sur cet identificateur de test donne
219.182.165.53.139.219.182.165.58.2278 (identificateur solution I1.I2.I3.I4).
Le filtre F1 échoue sur cet identificateur solution.

Le procédé continue en calculant la première instance potentielle

30 locale dont l'identificateur est strictement supérieur à l'identificateur solution.
Soit p le premier indice tel que I_p ne vérifie pas la condition $Cp_{(1)}$.

Ce n'est pas $p=1$ car il n'y a pas de contrainte sur $I1$;

C'est $p=2$ car $I2=139$ ne vérifie pas $C2_{(1)}$.

Le procédé rentre dans la boucle de calcul (le procédé reste dans la
5 boucle tant que $p>0$ et qu'aucune instance n'a été trouvée) :

$p=2$: existe-t-il un $J2_{(1)}$ supérieur à $I2=139$ qui vérifie $C2_{(1)}$? Non.

$p=1$: existe-t-il un $J1_{(1)}$ supérieur à $I1=219.182.165.53$? Oui :

$J1_{(1)}=219.182.165.54$

Le nouvel identificateur de l'instance potentielle est donc
10 $J1_{(1)}.I2_{(1)}.I3_{(1)}.I4_{(1)}$, soit $219.182.165.54.21.0.0.0.0.1024$. Le nouvel
identificateur de test est alors $219.182.165.54.21.0.0.0.0.1023$. Le résultat
du sixième GETNEXT sur cet identificateur de test indique qu'il n'y a plus
d'autre instance. On peut alors répondre à la requête complexe que la
recherche est finie.

15

Il est à noter que pour parcourir les 41 instances de la table selon l'art
antérieur, il aurait fallu 42 requêtes GETNEXT (il faut une requête
GETNEXT supplémentaire pour détecter la fin de la table), alors que selon
l'invention, 6 requêtes GETNEXT ont suffi.

20

La présente invention concerne donc un procédé de traitement d'une
requête complexe adressée à au moins un agent 5 SNMP de la machine 2b
ressource du système informatique 1 à partir du gestionnaire 4 de protocole
complexe de la machine 2a application, chaque agent 5 gérant des tables
25 d'attributs appartenant à la machine 2b ressource, les instances des tables
étant référencées par des identificateurs comprenant des index, caractérisé
en ce qu'il consiste à :

- transformer un filtre $F1$ issu de la requête complexe en un filtre $F2$
simplifié ne comportant que des conditions sur des index, le filtre $F2$
30 respectant les caractéristiques de correspondance suivantes : le filtre
 $F2$ laisse passer toutes les requêtes SNMP dont les réponses.

pourraient vérifier le filtre F1, mais filtre toutes les requêtes SNMP dont les réponses ne peuvent en aucun cas vérifier le filtre F1 ;

- restreindre les requêtes SNMP à celles qui respectent le filtre F2, et appliquer le filtre F1 sur les réponses.

5

Le procédé consiste à :

- 1) transformer le filtre F1 issu de la requête complexe en un filtre F2 simplifié ne comportant que des conditions sur des index et respectant les caractéristiques de correspondance ;
- 10 2) déterminer la première instance potentielle vérifiant le filtre F2 simplifié ; l'identificateur qui est juste inférieur à l'identificateur de l'instance potentielle déterminée est appelé identificateur de test ;
- 15 3) retrouver à partir d'une requête SNMP l'instance de la table ayant pour identificateur celui qui suit l'identificateur de test. Si aucune instance n'est retrouvée, le procédé de traitement est terminé. Si une instance est retrouvée, l'instance retrouvée est appelée instance solution ;
- 20 4) appliquer le filtre complexe F1 à l'instance solution ; si l'instance vérifie le filtre F1, elle fait partie de la réponse à la requête complexe traitée ;
- 25 5) déterminer la première instance potentielle dont l'identificateur est supérieur à l'identificateur de l'instance solution et vérifiant le filtre F2 simplifié. Si aucune instance n'est retrouvée, le procédé de traitement est terminé. Si une instance est retrouvée, l'identificateur qui est juste inférieur à l'identificateur de l'instance potentielle est appelé identificateur de test et le procédé reprend à la troisième étape.

Le procédé consiste à obtenir dans la première étape un filtre simplifié de la forme :

30

(OU

(ET



condition sur l'index 1 : $C1_{(1)}$

condition sur l'index 2 : $C2_{(1)}$

...

condition sur l'index n : $Cn_{(1)}$

5

)

...

(ET

condition sur l'index 1 : $C1_{(0)}$

condition sur l'index 2 : $C2_{(0)}$

10

...

condition sur l'index n : $Cn_{(0)}$

)

...

).

15

Si, dans la première étape, après simplification, le filtre se résume :

- à la seule condition VRAI, la table est parcourue dans son intégralité ;
- à la seule condition FAUX, aucune instance ne peut convenir.

20

Le procédé consiste, pour obtenir le filtre simplifié F2, à vérifier de prime abord si le filtre complexe répond à des règles définissant des filtres qui ne sont vérifiés par aucune instance.

Le procédé consiste, pour obtenir le filtre simplifié F2, à :

25

- transformer le filtre complexe en une combinaison de conditions sur les attributs liées par les opérateurs logiques ET, OU et NON ;
- repousser les opérateurs NON vers les feuilles et supprimer les NON doubles (NON NON) ;
- supprimer les conditions X portant sur des attributs qui ne sont pas des

30

- simplifier les opérations en résultant ;

- factoriser les ET et OU imbriqués ;
- regrouper les conditions concernant le même index ;
- regrouper tous les OU à la racine du filtre et simplifier à nouveau.

5 Le procédé consiste, pour supprimer les conditions X, à remplacer les conditions X et NON X par la constante VRAI.

Le procédé consiste, pour simplifier les opérations, à :

- remplacer les tests ET et OU à un seul opérande par cet opérande ;
- 10 • remplacer les opérations ET ne contenant que des opérandes VRAI par la constante VRAI et les opérations OU ne contenant que des opérandes FAUX par la constante FAUX ;
- retirer les conditions VRAI des autres opérations ET et les conditions FAUX des autres opérations OU ;
- 15 • remplacer les opérations OU contenant au moins un opérande VRAI par la constante VRAI et les opérations ET contenant au moins un opérande FAUX par la constante FAUX ;
- remplacer les conditions s'avérant être toujours VRAI ou FAUX par la constante VRAI ou FAUX ;
- 20 toutes ces opérations de simplification étant appliquées autant de fois qu'il est possible de le faire.

Le procédé consiste, dans la deuxième étape, à concaténer la première valeur vérifiant $C1_0$ à la première valeur vérifiant $C2_0$ jusqu'à Cn_0
 25 pour obtenir les instances potentielles locales zéro $I1_0, I2_0, \dots, In_0$, la première valeur possible en l'absence de condition sur un index donné étant la valeur nulle, l'instance potentielle correspondant à la plus petite des instances potentielles locales zéro.

Le procédé consiste, dans la cinquième étape, à effectuer pour tout i et tant que l'indice p est supérieur à 0 ou qu'aucune instance cherchée n'est trouvée, les opérations suivantes :

5 S'il existe un $J_{p_0} > I_p$ qui vérifie la condition C_{p_0} , alors l'instance potentielle locale est constituée de la manière suivante:

- pour tout index $k < p$, on prend la valeur I_k avec I1.I2. ...
In l'identificateur de l'instance solution ;
- pour l'index p , on prend la valeur J_{p_0} ;
- pour tout index $k > p$, on prend la valeur I_{k_0} ;

10 Sinon p prend la valeur $p-1$ et le procédé reprend les opérations ci-dessus, l'instance potentielle correspondant à la plus petite des instances potentielles locales obtenues.

15 Le procédé consiste, dans la deuxième et la cinquième étapes, à obtenir l'identificateur de test à partir de l'identificateur de l'instance potentielle, en soustrayant 1 à son dernier chiffre si celui-ci est différent de 0, ou en supprimant ce dernier chiffre s'il est nul.

20 Le procédé concerne également le système de traitement de la requête complexe adressée à au moins un agent 5 SNMP de la machine 2b ressource du système informatique 1 à partir du gestionnaire 4 de protocole complexe de la machine 2a application, chaque agent 5 gérant des tables d'attributs appartenant à la machine 2b ressource, les instances des tables étant référencées par des identificateurs comprenant des index, le système
25 comprenant l'agent intégrateur 6 permettant la mise en œuvre du procédé de traitement décrit précédemment.

ANNEXE 1

Exemple de valeurs de la table de connexion :

```

5  tcpConnState.0.0.0.0.0.0.0.0.0 = 1 .
   tcpConnState.0.0.0.0.7.0.0.0.0 = 2
   tcpConnState.0.0.0.0.9.0.0.0.0 = 2
   tcpConnState.0.0.0.0.13.0.0.0.0 = 2
   tcpConnState.0.0.0.0.19.0.0.0.0 = 2
   tcpConnState.0.0.0.0.21.0.0.0.0 = 2
10  tcpConnState.0.0.0.0.23.0.0.0.0 = 2
   tcpConnState.0.0.0.0.25.0.0.0.0 = 2
   tcpConnState.0.0.0.0.37.0.0.0.0 = 2
   tcpConnState.0.0.0.0.80.0.0.0.0 = 2
   tcpConnState.0.0.0.0.111.0.0.0.0 = 2
15  tcpConnState.0.0.0.0.139.0.0.0.0 = 2
   tcpConnState.0.0.0.0.199.0.0.0.0 = 2
   tcpConnState.0.0.0.0.512.0.0.0.0 = 2
   tcpConnState.0.0.0.0.513.0.0.0.0 = 2
   tcpConnState.0.0.0.0.514.0.0.0.0 = 2
20  tcpConnState.0.0.0.0.540.0.0.0.0 = 2
   tcpConnState.0.0.0.0.659.0.0.0.0 = 2
   tcpConnState.0.0.0.0.757.0.0.0.0 = 2
   tcpConnState.0.0.0.0.779.0.0.0.0 = 2
   tcpConnState.0.0.0.0.790.0.0.0.0 = 2
25  tcpConnState.0.0.0.0.793.0.0.0.0 = 2
   tcpConnState.0.0.0.0.919.0.0.0.0 = 2
   tcpConnState.0.0.0.0.924.0.0.0.0 = 2
   tcpConnState.0.0.0.0.1024.0.0.0.0 = 2
   tcpConnState.0.0.0.0.1025.0.0.0.0 = 2
30  tcpConnState.0.0.0.0.2401.0.0.0.0 = 2
   tcpConnState.0.0.0.0.2600.0.0.0.0 = 2
   tcpConnState.0.0.0.0.6000.0.0.0.0 = 2
   tcpConnState.127.0.0.1.1026.127.0.0.1.2600 = 5
   tcpConnState.127.0.0.1.2600.127.0.0.1.1026 = 5
35  tcpConnState.219.182.165.53.23.219.182.165.55.4109 = 5
   tcpConnState.219.182.165.53.139.219.182.165.58.2278 = 5
   tcpConnState.219.182.165.53.1017.219.182.100.2.1018 = 5
   tcpConnState.219.182.165.53.1018.219.182.100.2.514 = 7
   tcpConnState.219.182.165.53.1020.219.182.165.55.513 = 5
40  tcpConnState.219.182.165.53.1021.219.182.165.55.513 = 5
   tcpConnState.219.182.165.53.1022.219.182.100.2.1019 = 5
   tcpConnState.219.182.165.53.1023.219.182.100.2.514 = 7
   tcpConnState.219.182.165.53.1905.219.182.165.55.21 = 8
   tcpConnState.219.182.165.53.6000.219.182.100.2.1304 = 5
45  tcpConnLocalAddress.0.0.0.0.0.0.0.0.0 = 0.0.0.0
   tcpConnLocalAddress.0.0.0.0.7.0.0.0.0 = 0.0.0.0
   tcpConnLocalAddress.0.0.0.0.9.0.0.0.0 = 0.0.0.0
   tcpConnLocalAddress.0.0.0.0.13.0.0.0.0 = 0.0.0.0
   tcpConnLocalAddress.0.0.0.0.19.0.0.0.0 = 0.0.0.0
50  tcpConnLocalAddress.0.0.0.0.21.0.0.0.0 = 0.0.0.0
   tcpConnLocalAddress.0.0.0.0.23.0.0.0.0 = 0.0.0.0
   tcpConnLocalAddress.0.0.0.0.25.0.0.0.0 = 0.0.0.0
   tcpConnLocalAddress.0.0.0.0.37.0.0.0.0 = 0.0.0.0
   tcpConnLocalAddress.0.0.0.0.80.0.0.0.0 = 0.0.0.0
55  tcpConnLocalAddress.0.0.0.0.111.0.0.0.0 = 0.0.0.0

```

```

tcpConnLocalAddress.0.0.0.0.139.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.199.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.512.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.513.0.0.0.0 = 0.0.0.0
5 tcpConnLocalAddress.0.0.0.0.514.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.540.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.659.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.757.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.779.0.0.0.0 = 0.0.0.0
10 tcpConnLocalAddress.0.0.0.0.790.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.793.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.919.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.924.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.1024.0.0.0.0 = 0.0.0.0
15 tcpConnLocalAddress.0.0.0.0.1025.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.2401.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.2600.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.0.0.0.0.6000.0.0.0.0 = 0.0.0.0
tcpConnLocalAddress.127.0.0.1.1026.127.0.0.1.2600 = 127.0.0.1
20 tcpConnLocalAddress.127.0.0.1.2600.127.0.0.1.1026 = 127.0.0.1
tcpConnLocalAddress.219.182.165.53.23.219.182.165.55.4109 = 219.182.165.53
tcpConnLocalAddress.219.182.165.53.139.219.182.165.58.2278 = 219.182.165.53
tcpConnLocalAddress.219.182.165.53.1017.219.182.100.2.1018 = 219.182.165.53
tcpConnLocalAddress.219.182.165.53.1018.219.182.100.2.514 = 219.182.165.53
25 tcpConnLocalAddress.219.182.165.53.1020.219.182.165.55.513 = 219.182.165.53
tcpConnLocalAddress.219.182.165.53.1021.219.182.165.55.513 = 219.182.165.53
tcpConnLocalAddress.219.182.165.53.1022.219.182.100.2.1019 = 219.182.165.53
tcpConnLocalAddress.219.182.165.53.1023.219.182.100.2.514 = 219.182.165.53
tcpConnLocalAddress.219.182.165.53.1905.219.182.165.55.21 = 219.182.165.53
30 tcpConnLocalAddress.219.182.165.53.6000.219.182.100.2.1304 = 219.182.165.53
tcpConnLocalPort.0.0.0.0.0.0.0.0.0 = 0
tcpConnLocalPort.0.0.0.0.7.0.0.0.0 = 7
tcpConnLocalPort.0.0.0.0.9.0.0.0.0 = 9
tcpConnLocalPort.0.0.0.0.13.0.0.0.0 = 13
35 tcpConnLocalPort.0.0.0.0.19.0.0.0.0 = 19
tcpConnLocalPort.0.0.0.0.21.0.0.0.0 = 21
tcpConnLocalPort.0.0.0.0.23.0.0.0.0 = 23
tcpConnLocalPort.0.0.0.0.25.0.0.0.0 = 25
tcpConnLocalPort.0.0.0.0.37.0.0.0.0 = 37
40 tcpConnLocalPort.0.0.0.0.80.0.0.0.0 = 80
tcpConnLocalPort.0.0.0.0.111.0.0.0.0 = 111
tcpConnLocalPort.0.0.0.0.139.0.0.0.0 = 139
tcpConnLocalPort.0.0.0.0.199.0.0.0.0 = 199
tcpConnLocalPort.0.0.0.0.512.0.0.0.0 = 512
45 tcpConnLocalPort.0.0.0.0.513.0.0.0.0 = 513
tcpConnLocalPort.0.0.0.0.514.0.0.0.0 = 514
tcpConnLocalPort.0.0.0.0.540.0.0.0.0 = 540
tcpConnLocalPort.0.0.0.0.659.0.0.0.0 = 659
tcpConnLocalPort.0.0.0.0.757.0.0.0.0 = 757
50 tcpConnLocalPort.0.0.0.0.779.0.0.0.0 = 779
tcpConnLocalPort.0.0.0.0.790.0.0.0.0 = 790
tcpConnLocalPort.0.0.0.0.793.0.0.0.0 = 793
tcpConnLocalPort.0.0.0.0.919.0.0.0.0 = 919
tcpConnLocalPort.0.0.0.0.924.0.0.0.0 = 924
55 tcpConnLocalPort.0.0.0.0.1024.0.0.0.0 = 1024
tcpConnLocalPort.0.0.0.0.1025.0.0.0.0 = 1025
tcpConnLocalPort.0.0.0.0.2401.0.0.0.0 = 2401
tcpConnLocalPort.0.0.0.0.2600.0.0.0.0 = 2600

```

```

tcpConnLocalPort.0.0.0.0.6000.0.0.0.0 = 6000
tcpConnLocalPort.127.0.0.1.1026.127.0.0.1.2600 = 1026
tcpConnLocalPort.127.0.0.1.2600.127.0.0.1.1026 = 2600
tcpConnLocalPort.219.182.165.53.23.219.182.165.55.4109 = 23
5 tcpConnLocalPort.219.182.165.53.139.219.182.165.58.2278 = 139
tcpConnLocalPort.219.182.165.53.1017.219.182.100.2.1018 = 1017
tcpConnLocalPort.219.182.165.53.1018.219.182.100.2.514 = 1018
tcpConnLocalPort.219.182.165.53.1020.219.182.165.55.513 = 1020
tcpConnLocalPort.219.182.165.53.1021.219.182.165.55.513 = 1021
10 tcpConnLocalPort.219.182.165.53.1022.219.182.100.2.1019 = 1022
tcpConnLocalPort.219.182.165.53.1023.219.182.100.2.514 = 1023
tcpConnLocalPort.219.182.165.53.1905.219.182.165.55.21 = 1905
tcpConnLocalPort.219.182.165.53.6000.219.182.100.2.1304 = 6000
tcpConnRemAddress.0.0.0.0.0.0.0.0 = 0.0.0.0
15 tcpConnRemAddress.0.0.0.0.7.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.9.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.13.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.19.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.21.0.0.0.0 = 0.0.0.0
20 tcpConnRemAddress.0.0.0.0.23.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.25.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.37.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.80.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.111.0.0.0.0 = 0.0.0.0
25 tcpConnRemAddress.0.0.0.0.139.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.199.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.512.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.513.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.514.0.0.0.0 = 0.0.0.0
30 tcpConnRemAddress.0.0.0.0.540.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.659.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.757.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.779.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.790.0.0.0.0 = 0.0.0.0
35 tcpConnRemAddress.0.0.0.0.793.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.919.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.924.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.1024.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.1025.0.0.0.0 = 0.0.0.0
40 tcpConnRemAddress.0.0.0.0.2401.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.2600.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.0.0.0.0.6000.0.0.0.0 = 0.0.0.0
tcpConnRemAddress.127.0.0.1.1026.127.0.0.1.2600 = 127.0.0.1
tcpConnRemAddress.127.0.0.1.2600.127.0.0.1.1026 = 127.0.0.1
45 tcpConnRemAddress.219.182.165.53.23.219.182.165.55.4109 = 219.182.165.55
tcpConnRemAddress.219.182.165.53.139.219.182.165.58.2278 = 219.182.165.58
tcpConnRemAddress.219.182.165.53.1017.219.182.100.2.1018 = 219.182.100.2
tcpConnRemAddress.219.182.165.53.1018.219.182.100.2.514 = 219.182.100.2
tcpConnRemAddress.219.182.165.53.1020.219.182.165.55.513 = 219.182.165.55
50 tcpConnRemAddress.219.182.165.53.1021.219.182.165.55.513 = 219.182.165.55
tcpConnRemAddress.219.182.165.53.1022.219.182.100.2.1019 = 219.182.100.2
tcpConnRemAddress.219.182.165.53.1023.219.182.100.2.514 = 219.182.100.2
tcpConnRemAddress.219.182.165.53.1905.219.182.165.55.21 = 219.182.165.55
tcpConnRemAddress.219.182.165.53.6000.219.182.100.2.1304 = 219.182.100.2
55 tcpConnRemPort.0.0.0.0.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.7.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.9.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.13.0.0.0.0 = 0

```

```

tcpConnRemPort.0.0.0.0.19.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.21.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.23.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.25.0.0.0.0 = 0
5 tcpConnRemPort.0.0.0.0.37.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.80.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.111.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.139.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.199.0.0.0.0 = 0
10 tcpConnRemPort.0.0.0.0.512.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.513.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.514.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.540.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.659.0.0.0.0 = 0
15 tcpConnRemPort.0.0.0.0.757.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.779.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.790.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.793.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.919.0.0.0.0 = 0
20 tcpConnRemPort.0.0.0.0.924.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.1024.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.1025.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.2401.0.0.0.0 = 0
tcpConnRemPort.0.0.0.0.2600.0.0.0.0 = 0
25 tcpConnRemPort.0.0.0.0.6000.0.0.0.0 = 0
tcpConnRemPort.127.0.0.1.1026.127.0.0.1.2600 = 2600
tcpConnRemPort.127.0.0.1.2600.127.0.0.1.1026 = 1026
tcpConnRemPort.219.182.165.53.23.219.182.165.55.4109 = 4109
tcpConnRemPort.219.182.165.53.139.219.182.165.58.2278 = 2278
30 tcpConnRemPort.219.182.165.53.1017.219.182.100.2.1018 = 1018
tcpConnRemPort.219.182.165.53.1018.219.182.100.2.514 = 514
tcpConnRemPort.219.182.165.53.1020.219.182.165.55.513 = 513
tcpConnRemPort.219.182.165.53.1021.219.182.165.55.513 = 513
tcpConnRemPort.219.182.165.53.1022.219.182.100.2.1019 = 1019
35 tcpConnRemPort.219.182.165.53.1023.219.182.100.2.514 = 514
tcpConnRemPort.219.182.165.53.1905.219.182.165.55.21 = 21
tcpConnRemPort.219.182.165.53.6000.219.182.100.2.1304 = 1304

```


REVENDECATIONS

1. Procédé de traitement d'une requête complexe adressée à au moins un
5 agent (5) SNMP d'une machine (2b) ressource d'un système informatique (1)
à partir d'un gestionnaire (4) de protocole complexe d'une machine (2a)
application, chaque agent (5) gérant des tables d'attributs appartenant à la
machine (2b) ressource, les instances des tables étant référencées par des
identificateurs comprenant des index, caractérisé en ce qu'il consiste à :
- 10 • transformer un filtre (F1) issu de la requête complexe en un filtre (F2)
simplifié ne comportant que des conditions sur des index, le filtre (F2)
respectant les caractéristiques de correspondance suivantes : le filtre
(F2) laisse passer toutes les requêtes SNMP dont les réponses
pourraient vérifier le filtre (F1), mais filtre toutes les requêtes SNMP
15 dont les réponses ne peuvent en aucun cas vérifier le filtre (F1) ;
- restreindre les requêtes SNMP à celles qui respectent le filtre (F2), et
appliquer le filtre (F1) sur les réponses.
2. Procédé selon la revendication 1, caractérisé en ce qu'il consiste à :
- 20 1) transformer le filtre (F1) issu de la requête complexe en filtre (F2)
simplifié ;
- 2) déterminer la première instance potentielle vérifiant le filtre (F2)
simplifié ; l'identificateur qui est juste inférieur à l'identificateur de
l'instance potentielle déterminée est appelé identificateur de test ;
- 25 3) retrouver à partir d'une requête SNMP l'instance de la table ayant
pour identificateur celui qui suit l'identificateur de test. Si aucune
instance n'est retrouvée, le procédé de traitement est terminé. Si une
instance est retrouvée, l'instance retrouvée est appelée instance
solution ;
- 30 4) appliquer le filtre complexe (F1) à l'instance solution ; si l'instance
vérifie le filtre (F1), elle fait partie de la réponse à la requête
complexe traitée ;

5) déterminer la première instance potentielle dont l'identificateur est supérieur à l'identificateur de l'instance solution et vérifiant le filtre (F2) simplifié. Si aucune instance n'est retrouvée, le procédé de traitement est terminé. Si une instance est retrouvée, l'identificateur qui est juste inférieur à l'identificateur de l'instance potentielle est appelé identificateur de test et le procédé reprend à la troisième étape.

3. Procédé selon la revendication 2, caractérisé en ce qu'il consiste à obtenir dans la première étape un filtre simplifié de la forme :

(OU

(ET

condition sur l'index 1 : $C1_{(1)}$

condition sur l'index 2 : $C2_{(1)}$

...

condition sur l'index n : $Cn_{(1)}$

)

...

(ET

condition sur l'index 1 : $C1_{(0)}$

condition sur l'index 2 : $C2_{(0)}$

...

condition sur l'index n : $Cn_{(0)}$

)

...

).

4. Procédé selon l'une des revendications 2 ou 3, caractérisé en ce que si, dans la première étape, après simplification, le filtre se résume :

- à la seule condition VRAI, la table est parcourue dans son intégralité ;
- à la seule condition FAUX, aucune instance ne peut convenir.

5. Procédé selon l'une des revendications 2 à 4, caractérisé en ce qu'il consiste, pour obtenir le filtre simplifié F2, à vérifier de prime abord si le filtre complexe répond à des règles définissant des filtres qui ne sont vérifiés par
5 aucune instance.

6. Procédé selon l'une des revendications 1 à 5, caractérisé en ce qu'il consiste, pour obtenir le filtre simplifié F2, à :

- 10 • transformer le filtre complexe en une combinaison de conditions sur les attributs liées par les opérateurs logiques ET, OU et NON ;
- repousser les opérateurs NON vers les feuilles et supprimer les NON doubles (NON NON) ;
- supprimer les conditions X portant sur des attributs qui ne sont pas des index ;
- 15 • simplifier les opérations en résultant ;
- factoriser les ET et OU imbriqués ;
- regrouper les conditions concernant le même index ;
- regrouper tous les OU à la racine du filtre et simplifier à nouveau.

20 7. Procédé selon la revendication 6, caractérisé en ce qu'il consiste, pour supprimer les conditions X, à remplacer les conditions X et NON X par la constante VRAI.

25 8. Procédé selon l'une des revendications 6 ou 7, caractérisé en ce qu'il consiste, pour simplifier les opérations, à :

- remplacer les tests ET et OU à un seul opérande par cet opérande ;
- remplacer les opérations ET ne contenant que des opérandes VRAI par la constante VRAI et les opérations OU ne contenant que des opérandes FAUX par la constante FAUX ;
- 30 • retirer les conditions VRAI des autres opérations ET et les conditions FAUX des autres opérations OU ;

- remplacer les opérations OU contenant au moins un opérande VRAI par la constante VRAI et les opérations ET contenant au moins un opérande FAUX par la constante FAUX ;
- remplacer les conditions s'avérant être toujours VRAI ou FAUX par la constante VRAI ou FAUX ;

5

toutes ces opérations de simplification étant appliquées autant de fois qu'il est possible de le faire.

9. Procédé selon l'une des revendications 2 à 8, caractérisé en ce qu'il consiste, dans la deuxième étape, à concaténer la première valeur vérifiant $C1_0$ à la première valeur vérifiant $C2_0$ jusqu'à Cn_0 pour obtenir les instances potentielles locales zéro $I1_0, I2_0, \dots, In_0$, la première valeur possible en l'absence de condition sur un index donné étant la valeur nulle, l'instance potentielle correspondant à la plus petite des instances
- 10
- 15
- potentielles locales zéro.

10. Procédé selon la revendication 9, caractérisé en ce qu'il consiste, dans la cinquième étape, à effectuer pour tout i et tant que l'indice p est supérieur à 0 ou qu'aucune instance cherchée n'est trouvée, les opérations suivantes :

- 20
- S'il existe un $Jp_0 > Ip$ qui vérifie la condition Cp_0 , alors l'instance potentielle locale est constituée de la manière suivante:

- pour tout index $k < p$, on prend la valeur Ik avec $I1, I2, \dots$.
In l'identificateur de l'instance solution ;

- pour l'index p , on prend la valeur Jp_0 ;

- 25
- pour tout index $k > p$, on prend la valeur Ik_0 ;

Sinon p prend la valeur $p-1$ et le procédé reprend les opérations ci-dessus, l'instance potentielle correspondant à la plus petite des instances potentielles locales obtenues.

- 30
11. Procédé selon l'une des revendications 2 à 10, caractérisé en ce qu'il consiste, dans la deuxième et la cinquième étapes, à obtenir l'identificateur

de test à partir de l'identificateur de l'instance potentielle, en soustrayant 1 à son dernier chiffre si celui-ci est différent de 0, ou en supprimant ce dernier chiffre s'il est nul.

- 5 12. Système de traitement d'une requête complexe adressée à au moins un agent (5) SNMP d'une machine (2b) ressource d'un système informatique (1) à partir d'un gestionnaire (4) de protocole complexe d'une machine (2a) application, chaque agent (5) gérant des tables d'attributs appartenant à la machine (2b) ressource, les instances des tables étant référencées par des
- 10 identificateurs comprenant des index, le système comprenant un agent intégrateur (6) permettant la mise en œuvre du procédé de traitement selon l'une des revendications 1 à 11.

REVENDEICATIONS

1. Procédé de traitement d'une requête complexe adressée à au moins un
5 agent (5) SNMP d'une machine (2b) ressource d'un système informatique (1)
à partir d'un gestionnaire (4) de protocole complexe d'une machine (2a)
application, les machines (2a) application et (2b) ressource communiquant
au travers d'un réseau (3), chaque agent (5) gérant des tables d'attributs
appartenant à la machine (2b) ressource, les instances des tables étant
10 référencées par des identificateurs comprenant des index, caractérisé en ce
qu'il consiste à :
- transformer un filtre (F1) issu de la requête complexe du gestionnaire
(4) de la machine (2a) application en un filtre (F2) simplifié ne
comportant que des conditions sur des index, le filtre (F2) respectant
15 les caractéristiques de correspondance suivantes : le filtre (F2) laisse
passer toutes les requêtes SNMP dont les réponses pourraient
vérifier le filtre (F1), mais filtre toutes les requêtes SNMP dont les
réponses ne peuvent en aucun cas vérifier le filtre (F1) ;
 - restreindre les requêtes SNMP à celles qui respectent le filtre (F2) ;
 - 20 • transmettre lesdites requêtes SNMP restreintes à l'agent (5) SNMP
de la machine (2b) ressource au travers du réseau (3) ;
 - appliquer le filtre (F1) sur les réponses obtenues aux requêtes
SNMP ;
- le procédé permettant de traiter ladite requête complexe et d'optimiser le
25 nombre de requêtes SNMP transmises sur le réseau (3).
2. Procédé selon la revendication 1, caractérisé en ce qu'il consiste à :
- 1) transformer le filtre (F1) issu de la requête complexe en filtre (F2)
simplifié ;

- 2) déterminer la première instance potentielle vérifiant le filtre (F2) simplifié ; l'identificateur qui est juste inférieur à l'identificateur de l'instance potentielle déterminée est appelé identificateur de test ;
- 3) retrouver à partir d'une requête SNMP l'instance de la table ayant pour identificateur celui qui suit l'identificateur de test. Si aucune instance n'est retrouvée, le procédé de traitement est terminé. Si une instance est retrouvée, l'instance retrouvée est appelée instance solution ;
- 4) appliquer le filtre complexe (F1) à l'instance solution ; si l'instance vérifie le filtre (F1), elle fait partie de la réponse à la requête complexe traitée ;
- 5) déterminer la première instance potentielle dont l'identificateur est supérieur à l'identificateur de l'instance solution et vérifiant le filtre (F2) simplifié. Si aucune instance n'est retrouvée, le procédé de traitement est terminé. Si une instance est retrouvée, l'identificateur qui est juste inférieur à l'identificateur de l'instance potentielle est appelé identificateur de test et le procédé reprend à la troisième étape.

3. Procédé selon la revendication 2, caractérisé en ce qu'il consiste à obtenir dans la première étape un filtre simplifié de la forme :

(OU

(ET

condition sur l'index 1 : $C1_{(1)}$

condition sur l'index 2 : $C2_{(1)}$

...

condition sur l'index n : $Cn_{(1)}$

)

...

(ET

condition sur l'index 1 : $C1_{(0)}$

condition sur l'index 2 : $C2_0$

...

condition sur l'index n : Cn_0

)

5

...

).

4. Procédé selon l'une des revendications 2 ou 3, caractérisé en ce que si, dans la première étape, après simplification, le filtre se résume :

- 10
- à la seule condition VRAI, la table est parcourue dans son intégralité ;
 - à la seule condition FAUX, aucune instance ne peut convenir.

5. Procédé selon l'une des revendications 2 à 4, caractérisé en ce qu'il consiste, pour obtenir le filtre simplifié F2, à vérifier de prime abord si le filtre

15 complexe répond à des règles définissant des filtres qui ne sont vérifiés par aucune instance.

6. Procédé selon l'une des revendications 1 à 5, caractérisé en ce qu'il consiste, pour obtenir le filtre simplifié F2, à :

- 20
- transformer le filtre complexe en une combinaison de conditions sur les attributs liées par les opérateurs logiques ET, OU et NON ;
 - repousser les opérateurs NON vers les feuilles et supprimer les NON doubles (NON NON) ;
 - supprimer les conditions X portant sur des attributs qui ne sont pas des
- 25 index ;
- simplifier les opérations en résultant ;
 - factoriser les ET et OU imbriqués ;
 - regrouper les conditions concernant le même index ;
 - regrouper tous les OU à la racine du filtre et simplifier à nouveau.
- 30

7. Procédé selon la revendication 6, caractérisé en ce qu'il consiste, pour supprimer les conditions X, à remplacer les conditions X et NON X par la constante VRAI.

5 8. Procédé selon l'une des revendications 6 ou 7, caractérisé en ce qu'il consiste, pour simplifier les opérations, à :

- remplacer les tests ET et OU à un seul opérande par cet opérande ;
- remplacer les opérations ET ne contenant que des opérandes VRAI par la constante VRAI et les opérations OU ne contenant que des
- 10 opérandes FAUX par la constante FAUX ;
- retirer les conditions VRAI des autres opérations ET et les conditions FAUX des autres opérations OU ;
- remplacer les opérations OU contenant au moins un opérande VRAI par la constante VRAI et les opérations ET contenant au moins un
- 15 opérande FAUX par la constante FAUX ;
- remplacer les conditions s'avérant être toujours VRAI ou FAUX par la constante VRAI ou FAUX ;

toutes ces opérations de simplification étant appliquées autant de fois qu'il est possible de le faire.

20

9. Procédé selon l'une des revendications 2 à 8, caractérisé en ce qu'il consiste, dans la deuxième étape, à concaténer la première valeur vérifiant $C1_{(i)}$ à la première valeur vérifiant $C2_{(i)}$ jusqu'à $Cn_{(i)}$ pour obtenir les instances potentielles locales zéro $I1_0_{(i)}$, $I2_0_{(i)}$, ... $In_0_{(i)}$, la première

25 valeur possible en l'absence de condition sur un index donné étant la valeur nulle, l'instance potentielle correspondant à la plus petite des instances potentielles locales zéro.

10. Procédé selon la revendication 9, caractérisé en ce qu'il consiste, dans

30 la cinquième étape, à effectuer pour tout i et tant que l'indice p est supérieur à 0 ou qu'aucune instance cherchée n'est trouvée, les opérations suivantes :

S'il existe un $Jp_0 > Ip$ qui vérifie la condition Cp_0 , alors l'instance potentielle locale est constituée de la manière suivante:

- pour tout index $k < p$, on prend la valeur Ik avec 11.12. ...
In l'identificateur de l'instance solution ;
- pour l'index p , on prend la valeur Jp_0 ;
- pour tout index $k > p$, on prend la valeur Ik_{0_0} ;

Sinon p prend la valeur $p-1$ et le procédé reprend les opérations ci-dessus, l'instance potentielle correspondant à la plus petite des instances potentielles locales obtenues.

10

11. Procédé selon l'une des revendications 2 à 10, caractérisé en ce qu'il consiste, dans la deuxième et la cinquième étapes, à obtenir l'identificateur de test à partir de l'identificateur de l'instance potentielle, en soustrayant 1 à son dernier chiffre si celui-ci est différent de 0, ou en supprimant ce dernier chiffre s'il est nul.

15

12. Système de traitement d'une requête complexe adressée à au moins un agent (5) SNMP d'une machine (2b) ressource d'un système informatique (1) à partir d'un gestionnaire (4) de protocole complexe d'une machine (2a) application, chaque agent (5) gérant des tables d'attributs appartenant à la machine (2b) ressource, les instances des tables étant référencées par des identificateurs comprenant des index, le système comprenant un agent intégrateur (6) permettant la mise en œuvre du procédé de traitement selon l'une des revendications 1 à 11.

20

1/2

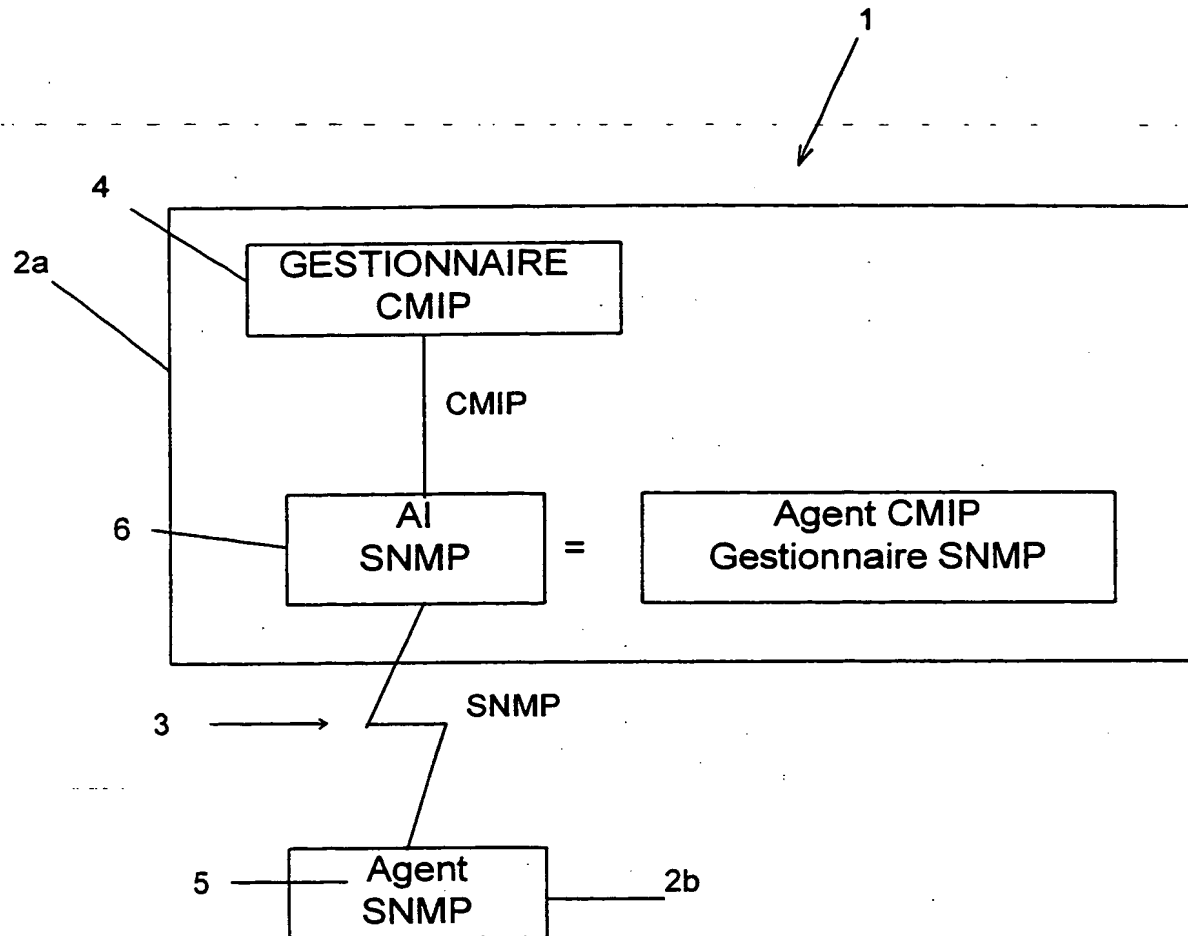


FIG.1

2/2

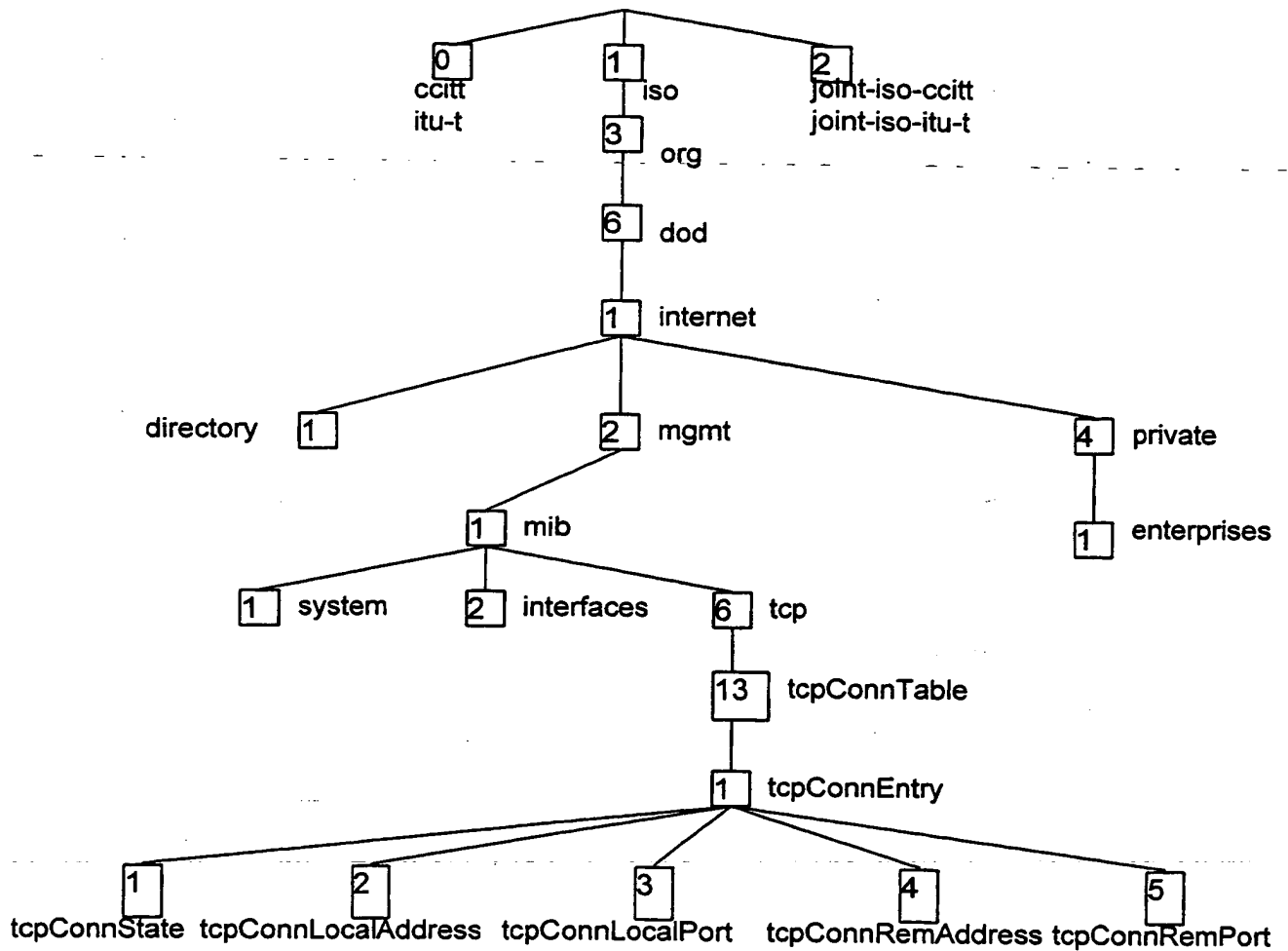


FIG.2

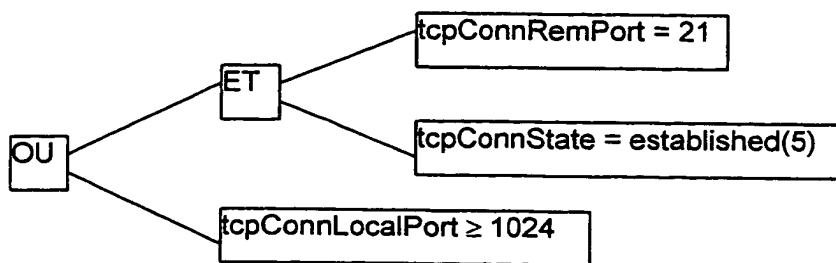


FIG.3

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☒ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

THIS PAGE BLANK (USPTO)